CLAIMS

1. A computer implemented method for rearranging a computer program comprising:

organizing the computer program into a plurality of blocks;

determining a critical section of the computer program;

constructing a dependency graph based on the organization of the computer program;

recognizing a portion of the computer program that could be executed outside of the critical section; and

inserting a plurality of dependency relationships between the plurality of blocks to cause execution of the recognized portion of the computer program outside of the critical section.

2. The method of claim 1, wherein a block includes a computer program instruction.

3. The method of claim 1 further comprises organizing the computer program based on a node and a super block, wherein the node includes a plurality of blocks and the super block includes a plurality of nodes.

4. The method of claim 1, wherein the critical section of the computer program accesses shared resources.

5. The method of claim 1 further comprises comprising determining to the extent the critical section is part of the dependency graph.

6. The method of claim 5 further comprises comprising adding a termination point to the critical section if a portion of the critical section is outside of the dependency graph.

7. The method of claim 1 further comprises comprising inserting additional dependency relationship based on a direct dependency, an indirect dependency, or a shortest life-time dependency.

8. The method of claim 1 further comprises comprising scheduling to execute the computer program based on the dependency graph.

9. A computer implemented system for rearranging a computer program comprising:

a computer program organizer, wherein the organizer organizes the computer program into a plurality of blocks;

a critical section determination module;

a dependency graph construction module, wherein a dependency graph is constructed based on the organization of the computer program; and

a dependency relationships inserter, wherein the dependency relationship is inserted between the plurality of blocks to cause execution of the recognized portion of the computer program outside of a critical section.

10. The system of claim 9, wherein the critical section determination module determines to the extent the critical section is part of the dependency graph.

11. The system of claim 9, wherein the critical section of the computer program accesses shared resources.

12. The system of claim 11, wherein the dependency relationships inserter inserts a termination point to the critical section if a portion of the critical section is outside of the dependency graph.

13. A system for processing a plurality of network packets comprising:

a network processor;

a network interface to control the transmission between the network processor and a network;

a shared resource accessible to the plurality of network packets;

a network processor program to process the plurality of network packets;

a dependency graph constructor to construct a dependency graph based on the network processor program; and

a dependency relationship inserter to optimize the network processor program by inserting a plurality of dependency relationships to rearrange the order in which the network processor program is executed.

14. The system in claim 13, wherein the dependency graph constructor further determines a critical section and to the extent a critical section is part of the dependency graph.

15. The system in claim 13, wherein the dependency relationship inserter module inserts additional dependency relationships based on a direct dependency, an indirect dependency, or a shortest life-time dependency.

16. A machine-accessible medium that provides instructions that, when executed by a processor, causes the processor to:

organize a computer program based on a plurality of blocks;

determine a critical section of the computer program;

construct a dependency graph based on the organization of the computer program;

recognize a portion of the computer program that could be executed outside of the critical section; and

insert a plurality of dependency relationships between the plurality of blocks to cause execution of the recognized portion of the computer program outside of the critical section.

17. The machine readable medium of method 16, wherein the critical section of the computer program accesses shared resources.

18. The machine readable medium of method 16 further comprises inserting a termination point to the critical section if a portion of the critical section is outside of the computer program.

19. The machine readable medium of method 16 further comprises inserting dependency relationships based on a direct dependency, an indirect dependency, or a shortest life-time dependency.